TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2012

END-YEAR

COMP 303
Design and Analysis of Algorithms

Time Allowed: Three Hours

Instructions:

- *Read each question carefully before attempting it.*
- This examination will be marked out of **180** marks.
- Answer all questions.
- You may answer the questions in any order. Make sure you clearly identify the question you are answering.
- Non-electronic foreign language-English dictionaries are permitted.
- Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Questions	Marks
1. Algorithm Analysis	[30]
2. Master Method	[10]
3. Computability and Complexity	[12]
4. Knapsacks	[30]
5. Graphs	[20]
6. Dynamic Programming	[30]
7. Divide and Conquer	[20]
8. Various Topics	[28]

The following definitions are provided for your convenience. You may find it useful to tear off this front page of the paper.

Asymptotic notation:

$$O(g(n)) = \{f(n) \mid (\exists d)(\mathbf{aa}\, n)[0 \leq f(n) \leq d \cdot g(n)]\}$$

$$\Omega(g(n)) = \{f(n) \mid (\exists c > 0)(\mathbf{aa}\, n)[f(n) \geq c \cdot g(n) \geq 0]\}$$

$$\Theta(g(n)) = \{f(n) \mid (\exists c > 0, d)(\mathbf{aa}\, n)[0 \leq c \cdot g(n) \leq f(n) \leq d \cdot g(n)]\}$$

Master Theorem: Let $T(n)$ be defined by the recurrence $T(n) = aT(n/b) + f(n)$.
Let $\alpha = \log_b a$.

1. If $(\exists \epsilon > 0)[f(n) \in O(n^{\alpha-\epsilon})]$ then $T(n) \in \Theta(n^{\alpha})$.
2. If $f(n) \in \Theta(n^{\alpha})$ then $T(n) \in \Theta(n^{\alpha} \log n)$.
3. If $(\exists \epsilon > 0)[f(n) \in \Omega(n^{\alpha+\epsilon})]$ and $(\exists c < 1)(\mathbf{aa}\, n)[a \cdot f(n/b) \leq c \cdot f(n)]$ then $T(n) \in \Theta(f(n))$.

Logarithms:

$\log_a x = y$ if and only if $a^y = x$
$\log_a x = \log_b x \div \log_b a$

Question 1. Algorithm Analysis [30 marks]

(a) Using the definitions for O, Ω, and Θ given on the front page of this paper, show that:

(i) [5 marks] $n^3 + n^2 \in O(n^3)$,

(ii) [5 marks] $\Omega(n^2) \subseteq \Omega(kn)$, for any positive integer k.

(b) This question concerns writing precise specifications.

(i) [5 marks] Write a specification of the problem of finding the smallest and largest elements of a sequence. The input to the problem is a sequence s, and the output is two values, *small* and *large*, where *small* is the smallest, and *large* is the largest.

(ii) [5 marks] Write pseudo-code describing an algorithm that uses a loop to solve the problem in subquestion 1(b)(i).

(iii) [5 marks] Your algorithm should contain a loop. Give a loop invariant that could be used to prove this algorithm is correct. (You do not need to provide a complete proof.)

(c) [5 marks] What is wrong with the following "proof" that any algorithm has a run time that is $O(n)$? Note that all of the text indented below is part of such "proof".

> We must show that the time required for an input of size n is at most a constant times n.
>
> **Basis Step.** Suppose that $n = 1$. If the algorithm takes C units of time for an input of size 1, the algorithm takes at most $C \times 1$ units of time. Thus, the assertion is true for $n = 1$.
>
> **Inductive Step.** Assume that the time required for an input of size n is at most $C' \times n$ and that the time for processing an additional item is C''. Let C be the maximum of C' and C''. Then the total time required for an input of size $n + 1$ is at most:
>
> $$C' \times n + C'' \leq C \times n + C = C \times (n + 1)$$
>
> The inductive step has been verified.
>
> By induction, for input of size n, the time required is at most a constant times n. Therefore, the run time is $O(n)$.

Question 2. Master Method [10 marks]

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 4T(n/3) + n \, logn$.
2. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Question 3. Computability and Complexity [12 marks]

(a) [4 marks] What is the difference between *a problem* and *an algorithm*?

(b) Define and explain in plain English the classes

(i) [2 marks] *P*,

(ii) [2 marks] *NP*,

(iii) [2 marks] *NP-Complete*, and

(iv) [2 marks] *NP-Hard*.

Question 4. Knapsacks [30 marks]

Discuss the applicability of each of the following algorithm design approaches to both the *Fractional Knapsack* problem and the *0-1 Knapsack* problem:

(a) [6 marks] Divide and Conquer,

(b) [8 marks] Dynamic Programming,

(c) [8 marks] Greedy Algorithms,

(d) [8 marks] Graph Algorithms.

Question 5. Graphs [20 marks]

We have a connected undirected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T.

(a) [10 marks] Prove that G is acyclic.

(b) [10 marks] Prove that $G = T$. In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.

Question 6. Dynamic Programming [30 marks]

Some time back, you helped a group of friends who were doing simulations for a computation-intensive investment company, and they have come back to you with a new problem. They are looking at n consecutive days of a given stock, at some point in the past. The days are numbered $i = 1, 2, ..., n$; for each day i, they have a price *p(i)* per share for the stock on that day.

For certain (possibly large) values of k, they want to study what they call *k-shot strategies*. A k-shot strategy is a collection of m pairs of days $(b_1, s_1), ..., (b_m, s_m)$, where $0 \leq m \leq k$ and

$$1 \leq b_1 < s_1 < b_2 < s_2 ... < b_m < s_m \leq n.$$

We view these as a set of up to k nonoverlapping intervals, during each of which the investors buy 1,000 shares of the stock (on day b_i) and then sell it (on day s_i). The *return* of a given k-shot strategy is simply the profit obtained from the m buy-sell transactions, namely,

$$1,000 \sum_{i=1}^{m} p(s_i) - p(b_i).$$

The investors want to assess the value of k-shot strategies by running simulations on their n-day trace of the stock price.

(a) [15 marks] Design and write down the pseudocode for an efficient algorithm that determines, given the sequence of prices, the k-shot strategy that would have had the maximum possible return. Since k may be relatively large in these simulations, your running time should be polynomial in both n and k; it should not contain k in the exponent.

(b) [5 marks] Show the asymptotic complexity of your algorithm.

(c) [10 marks] Show that your algorithm correctly gives the optimal answer.

Question 7. Divide and Conquer [20 marks]

Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labelled with a real number x_v. You may assume that the real numbers labelling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labelling is only specified in the following way: for each node v, you can determine the value x_v by *probing* the node v. Assume that this *probing* operation can be potentially very expensive. Show how to find a local minimum of T using at most $O(logn)$ *probes* to the nodes of T, starting at the root of the tree.

State your solution (presumably a Divide and Conquer algorithm) following the template from the lectures. *Prove your algorithm is correct by following the procedure shown in lectures.*

Question 8. Various Topics [28 marks]

(a) [4 marks] Describe a problem that would be suitable for a Monte Carlo algorithm and explain how randomness will help you make the algorithm more efficient.

(b) [4 marks] Describe a problem that would be suitable for a Las Vegas algorithm and explain how randomness will help you make the algorithm more efficient.

(c) [10 marks] Name a *nonblocking* data structure and explain how it works and what makes it *not block*.

(d) [10 marks] Prove that the problem of sorting n elements *based on comparisons* is $\Theta(n\, logn)$.

****